AD A030692

13

USERS GUIDE FOR LCPL

A Program for Solving Linear Complementarity Problems by Lemke's Method

BY

J. A. TOMLIN

TECHNICAL REPORT SOL 76-16

AUGUST 1976

See 1473

Systems Optimization Laboratory

Department of Operations Research

Stanford University

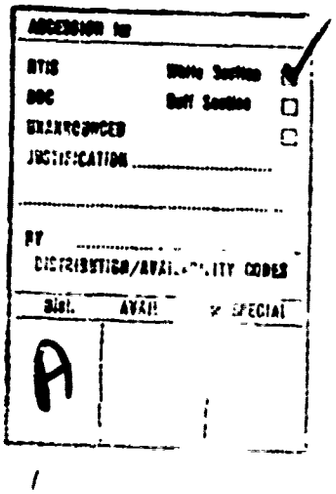
Stanford California 94305

DDC RECEIVED OCT 18 1976 D

USERS GUIDE FOR LCPL

A Program for Solving Linear Complementarity Problems

by Lemke's Method



by

J. A. Tomlin

TECHNICAL REPORT SOL 76-16

August 1976

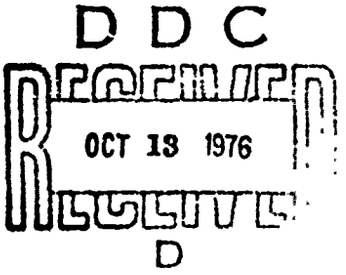
SYSTEMS OPTIMIZATION LABORATORY

DEPARTMENT OF OPERATIONS RESEARCH

Stanford University
Stanford, California

Research and reproduction of this report were partially supported by the U.S. Army Research Office-Durham, DAAC-29-74-C-0034; U.S. Energy Research and Development Administration Contract E(04-3)-326 PA #18; Office of Naval Research Contract N00014-75-C-0865; and National Science Foundation Grant DCR75-04544.

Reproduction in whole or in part is permitted for any purposes of the United States Government. This document has been approved for public release and sale; its distribution is unlimited.



CONTENTS

1. PRELIMINARIES 1

1.0. Introduction 1
1.1. The Linear Complementarity Problem 1
1.2. Lemke's Method 3
1.3. Implementation 3
1.4. Recovery Procedure 4
1.5. Quadratic Programming Objective 5

2. PROGRAM INPUT

2.0. Introduction 6
2.1. Parameters 6
2.2. Problem Input 11
2.3. Basis Input 13
2.4. Problem Size 14
2.5. Multiple Problems 14

3. PROGRAM OUTPUT 15

3.0. Introduction 15
3.1. Standard Output 15
3.2. Program Messages and Diagnostics 17

4. SAMPLE RUNS 23

4.0. Problem Descriptions 23
4.1. Program Input 24
4.2. Program Output 25
4.3. Use of the NQUAD Input Option 31

1. PRELIMINARIES

1.0. Introduction

LCPL is a program for solving Linear Complementarity Problems by Lemke's method. It supersedes the "NULEMKE" code issued by SOL in 1974. Considerable care has been taken to make the code as robust as possible. It employs more error checking and correction facilities, recovery procedures and more diagnostics than the earlier code. Scaling facilities are provided for badly formulated or ill-conditioned models. There is also a much larger and more flexible set of parameters which can be set by the user, though in general the default values will be satisfactory.

This document gives only a description of the use of the code in distributed form. The program itself, and the procedures for changing it to solve problems of larger dimension etc., are described in the "Programmers Guide for LCPL." We note here, however, that the program is all in FORTRAN H for IBM 360/370 computers and is WATFIV compatible.

1.1. The Linear Complementarity Problem

An LCP is a problem of the form:

Find w and z such that

$$w = q + Mz \ , \qquad w^T z = 0 \ , \qquad w, z \geq 0 \qquad (1)$$

In general M is required to have some special property, such as positive semi-definiteness, for the problem to have a solution, or more concretely, to be solvable by Lemke's method as implemented in LCPL.

Problems of type (1) arise in many contexts, perhaps the most common being quadratic programming. A solution of a semi-definite program:

$$\begin{aligned} &\min\ c^T x + \frac{1}{2} x^T D x \\ &\text{subject to } Ax \geq 0, \quad x \geq 0 \end{aligned} \tag{2}$$

may be obtained by solving for a Kuhn-Tucker point:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c \\ -b \end{bmatrix} + \begin{bmatrix} D & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3}$$

$$x^T u + y^T v = 0$$

$$x, y, u, v \geq 0$$

By means of the obvious correspondence

$$w = \begin{bmatrix} u \\ v \end{bmatrix}, \quad z = \begin{bmatrix} x \\ y \end{bmatrix}, \quad q = \begin{bmatrix} c \\ -b \end{bmatrix}, \quad M = \begin{bmatrix} D & -A^T \\ A & 0 \end{bmatrix}, \tag{4}$$

we have a problem of type (1). Note that if D is positive semi-definite, so is M. In the special case where D is zero the problem reduces to a linear program.

There are several other ways in which LCP's arise, some directly and some from equivalent problems. The user is referred to

R.W. Cottle and G.B. Dantzig, "Complementary Pivot Theory of Mathematical Programming," Linear Algebra and its Applications 1, 103-125 (1968).

1.2. Lemke's Method

Lemke's method solves problems of type (1) when M has certain properties. Several sufficient conditions are known, but a complete characterization is not available (see Cottle and Dantzig). Positive semi-definiteness of M, or the case where M has positive principal minors, are the best known sufficient conditions and cover a wide class of problems.

In outline, Lemke's method, as implemented in LCPL, appends a dummy column and variable to problem (1):

$$w = q + Mz + e\zeta \tag{5}$$

where e has +1's on those rows where q_i is negative (or on all rows). ζ is then pivoted immediately into the basis to make all the basic (w) variables non-negative. This eliminates some w_i, say w_p, from the basis and leads to a non-negative solution which is complementary for all but one pair. The complement of w_p, that is z_p, is then introduced into the basis leading to a new "almost complementary" solution. This is continued until either ζ leaves the basis or becomes zero, or an unbounded ray is encountered when attempting to introduce a variable into the basis (again see Cottle and Dantzig).

1.3. Implementation

Internally LCPL stores the problem in the form

$$Iw - Mz - e\zeta = q$$

that is with coefficient matrix [I, -M, -e]. Only the non-zero coefficients are stored in packed form. The vector e is constructed internally from the sign pattern of the given q for the problem (an optional parameter setting allows e to be a full column of + 1's if desired).

The basis inverse is maintained in standard product form, but uses an LU decomposition at INVERT time when a new inverse representation is computed. The inversion frequency is controlled by a parameter.

1.4. Recovery Procedure.

If a problem is ill-conditioned or some tolerances have been set to unsuitable values it is numerically possible for the algorithm to produce two error conditions: the basic solution becomes infeasible, that is contains a negative value, or the "basis" may be detected to be singular by INVERT. In either of these cases the following recovery procedure is carried out:

1) A nonsingular complementary basis $\bar{B}$, containing as many columns of M in common with the old basis as possible, is constructed (the e column is always removed).

(2) A transformed problem is (notationally) constructed

$$\bar{w} = \bar{q} + \bar{M}\bar{z}$$

where $\bar{q} = \bar{B}^{-1}q$, $\bar{M} = \bar{B}^{-1}M$ and $\bar{w}$ and $\bar{z}$ are a complementary permutation of the original w and z. Because of the construction

of $\bar{B}$ this is a principal transform of the original problem. Such a transform preserves solvability by Lemke's method if the original M is positive semi-definite or has positive principal minors.

3) A new column $\bar{e}$ is added which has a +1 on every row where $\bar{q}_i$ is negative. The transformed augmented problem:

$$\bar{w} = \bar{q} + \bar{M}\bar{z} + \bar{e}\zeta$$

is started in the same way as the original problem. (Note: Since the code keeps the original M and q throughout we wish to keep a representation of $\bar{e}$ in terms of the original unit basis. This is immediately to hand as $e' = \bar{B}\bar{e}$ which over-writes the original e).

1.5. Quadratic Programming Objective

Throughout the algorithm LCPL prints out the value of ζ ("DUMMY Z") at each iteration. If the problem derives from a quadratic program, set up as in (3), the user may specify the dimension of D (by the parameter NQUAD) and the value of $c^Tx + \frac{1}{2}x^TDx$ will be computed and printed on completion. In this case it is also unnecessary to supply the columns of $-A^T$.

2. PROGRAM INPUT

2.0. Introduction.

There are two parts to program input; the parameter list and the problem input. The parameters and their input is described in Section 2.1. The matrix M and q are input in a very slightly modified "MPS format" as used in linear programming. In addition we may specify a starting basis as specified in Section 2.3.

2.1. Parameters

Parameters are changed from their default value for each problem by means of a FORTRAN "NAMELIST" input from the card reader. The first card must start with &PARAM in column 2 or later; each card must begin with a blank, and the list must be terminated by &END.

Examples.

```
cols
1 2 3
   &P A R A M  NQUAD=64, IFSCAL=1, &END
   &PARAM  &END
```

The first example specifies that the problem is a quadratic program in the first 64 variables and calls for scaling of M. The second example leaves all parameters at their default values.

We group the parameters into the following groups:

Run Control:

ITRLIM (default = 99999)

Maximum number of iterations for this run of this problem.

INVFRQ (default = 30)

Inversion frequency

ITCH (default = INVFRQ)

Iteration frequency for checking the relative error in the current basic solution. Defaulted to do so just before normal inversion. (INVERT automatically carries out this check after inversion.)

IBFRQ (default = 99999)

Iteration frequency for writing the current basis to the file specified by KOUTB (not done unless KOUTB is specified to be non-zero). If KOUTB is properly specified the final basis will be saved regardless of the value of IBFRQ.

Input:

KINP (default = 5)

FORTRAN logical unit number for reading the problem.

IFNEG (default = 1)

Determines whether to change the sign of the input matrix M. With the default setting the user supplies the matrix M and -M is stored to collect all variables on the left-hand side as in (6). If the user supplies -M then set IFNEG = 0.

IFSCAL (default = 0)

This parameter is set to 1 if the matrix M is to be scaled. If scaling is performed the solution (and quadratic objective function value) are descaled to give their true values. Usually scaling will not be required.

IFALLE (default = 0)

If the parameter is set to 1 the dummy column e will be constructed with a 1 in every position. Not recommended for normal use since this increases the density of the transformations.

KINB (default = 0)

FORTRAN logical unit number for reading a basis. With the default value of 0 no basis reading will be attempted. If KINB is greater than 7 the unit is rewound before reading.

NQUAD (default = 0)

If NQUAD is non-zero the problem is assumed to be a quadratic program in the form (2), (3) in Section 1.1. In this case only the NQUAD columns of M corresponding to the quadratic variables x

need be supplied; the $-A^T$ column will be constructed from the last NQUAD rows of the column supplied. If M is non-square and NQUAD is not correctly specified a fatal error is generated. (See also output.)

Output:

IØUT (default = 0)

With the default setting all the w and z variables, names and values, are printed. If IØUT is set to a non-zero value only the basic variables are printed (in sorted order) together with the row names and the original right hand side q.

NQUAD (default = 0)

If NQUAD has some non-zero value n the first n rows and columns of M are assumed to be the Hessian of a quadratic form, as in (2). The value of the quadratic form will be printed out after the w and z variable values at solution time. (See also Input above.)

KOUTB (default = 0)

FORTRAN logical unit number for writing the basis every IBFRQ iterations and on termination. No basis is output if KOUTB is zero. Unit KOUTB is rewound before and after writing if KOUTB is greater than 7.

Tolerances:

ZTOLZE (default = 10^{-4})

Zero tolerance for testing feasibility. Basic variables are non-negative if they are $\geq$ -ZTOLZE. Complementarity is declared if ζ ("DUMMY Z") $\leq$ ZTOLZE.

ZTOLPV (default = 10^{-6})

Absolute pivot tolerance.

ZTOLRP (default = 10^{-8})

Relative pivot tolerance while iterating. A pivot is accepted if its absolute value is greater than the largest absolute value in the updated column multiplied by ZTOLRP. If this test fails INVERT is called and the same variable is considered again. If the chosen pivot element still fails the test it is accepted. Note that if ZTOLRP is large this could lead to inverting at every iteration. If ZTOLRP is too small the test becomes ineffective.

ZTETA (default = 10^{-12})

Tolerance on the absolute value of transformation elements. Larger values may give sparser transformations but at great cost to accuracy and stability.

ZTOLDA (default = 10^{-7})

Tolerance on the absolute value of entries in the matrix M. Any value smaller than this is disregarded.

2.2. Problem Input

The M and q defining the problem are read from unit KINP in slightly modified "MPS format." The card images required are:

NAME Card

This has "NAME" in columns 1 to 4 and the (up to 8 characters) problem name in columns 15-22. This card is optional but highly desirable (particularly for checking against basis names).

ROWS Card

Has "ROWS" in columns 1-4.

Row Names

Each row is assigned a (unique) name of up to 8 characters, one per card, in columns 5-12. Embedded blanks are allowed (unlike MPS). Note that there are no row types as in MPS and that any supplied will be ignored. Column 1 must be blank.

COLUMNS Card

Has "COLUMNS" in columns 1-7.

Matrix Element

The non-zero elements of M are supplied by column. All the elements of a column must be together. Each column is assigned a (unique) name up to eight characters. The format is:

cols	1 - 4	5 - 12	15 - 22	25 - 36	40 - 47	50 - 61
	blank	column name	row name	element value	second row name (optional)	second element value (optional)

Again embedded blanks are allowed in column names.

RHS Card

Has "RHS" in columns 1 - 3.

Right Hand Side Elements

The right hand side vector (q) may be given a name, which should be different from any row or column name. The elements are given in the same format as the matrix elements. Note that only one right hand side may be supplied. If an attempt is made to input more than one, the non-zeros from later right hand sides will over-write the earlier values.

ENDATA Card

Has "ENDATA" in columns 1 - 6.

Sample input is shown in section 4.1.

It is important to note that the "w" variables are assigned the row names of M, and the "z" variables the column names. The dummy column e is assigned the name "DUMMY Z", as is the dummy variable z.

2.3. Basis Input

A complementary, or almost complementary, basis may be supplied from unit KINB. For each complementary pair (w_i, z_i), where the z_i is basic, the name of z_i is given, otherwise w_i is assumed basic. If ζ ("DUMMY Z") is basic the name of the w_i variable in the non-complementary pair it replaces is given. The required cards and formats are:

NAME Card

Has "NAME" in columns 1 - 4 and the basis name in columns 15 - 22. If this name does not agree with the problem name, or the name card is missing, a warning is given.

Basis Cards

For basic columns of M the column names are given (one per card) in columns 5 - 12. If ζ is basic there must be a card with "DUMMY Z" in columns 5 - 12 and a row name in columns 15 - 22. This row name is the name of the w variable ζ replaces in the basis.

ENDATA Card

Has "ENDATA" in columns 1 - 6.

Note that if the basis supplied is singular or infeasible the program will enter the recovery procedure ("RECOVR"). If naming errors are encountered in reading the basis the program will start with the partial basis found (which will almost certainly trigger a call to RECOVR).

2.4. Problem Size

The standard version of LCPL is dimensioned to handle problems of up to 350 rows, with up to 4000 non-zero matrix elements in [I,-M,-e]. Up to 8000 eta (transformation) elements are allowed for. If these dimensions are inadequate the program must be changed (see the LCPL programmer's guide).

2.5. Multiple Problems

Any number of problems (each preceded by parameter cards) may be solved in sequence. Note that all parameters are reset to default values before each new set of parameters and problem is read.

3.0. Introduction

We may divide the program output into three categories. Standard problem description, inversion statistics, the iteration log and solution print are described in Section 3.1. Program messages and diagnostics are described in Section 3.2. The output format for a basis written on unit KOUTB is exactly the same as for basis output as described in Section 2.3.

3.1. Standard Output

First the values of all the user parameters are written in NAMELIST format for checking.

(a) Input Information: This includes the problem name and its statistics -number of rows, number of columns of M supplied, number of non-zero M elements and density. The number of non-zero (unit) elements in the dummy column e constructed is also given. See Section 4 for examples.

(b) Inversion Statistics: These include number of non-zeros in the basis, structural columns in the basis, the number of non-zeros in the L and U factors and the totals. The estimated relative error in the solution for the new inverse is also given (if this error is greater than 10^{-8} the inversion will be repeated with tighter tolerances).

(c) Iteration Log: Gives for each iteration:

ITCOUNT = iteration number

STATUS = { A almost complementary
C complementary
U unbounded almost complementary ray

DUMMY Z = value of the "dummy variable" ζ

VECIN = name of column entering basis

VECOUNT = name of column leaving basis

NETA = number of etas in the eta file

NELEM = number of elements in the eta file.

(d) Solution Print-Out: The standard option gives the names and values of all the "w" and "z" variables. If "DUMMY Z" is basic its value is printed after the "z" variables.

If IØUT = 1 only the basic variables are printed (in sorted order) together with the row names and the original q values.

If NQUAD is set $c^Tx + \frac{1}{2}x^TDx$ will be evaluated (see (2) in Section 1.1) and the message

QUADRATIC OBJECTIVE FUNCTION VALUE = xxxxx.xxxx

is printed.

See the sample run output for examples.

3.4. Program Messages and Diagnostics

The following informative messages may be printed, depending on parameter settings, problem status, etc.

Input

SPLIT VECTOR 'name'

A column with the same 'name' was encountered earlier. The input deck is probably corrupted. Input continues however.

NO MATCH FOR ROW 'rowname' AT COLUMN 'colname'

Column 'colname' contains a nonzero on an unidentifiable row 'rowname'. Check alignment of row names in rows and columns sections. This is a fatal error.

NUMBER OF MATRIX ELEMENTS 'number' EXCEEDS NAMAX-ABORT

The program dimensions for A(·) and IA(·) must be increased to solve the problem. Fatal error. Standard NAMAX is 4000. Message can also occur in RECOVR.

MATRIX NOT SQUARE AND NQUAD INCORRECTLY SPECIFIED-PROBLEM ABANDONED.

The number of structural columns supplied is not equal to either NROW or the specified value of NQUAD. M cannot therefore be properly constructed to be square. Fatal error.

NQUAD COLUMNS ONLY SUPPLIED-WILL ATTEMPT TO FORM 'number' EXTRA COLUMNS OF CONSTRAINT MATRIX TRANSPOSE

The problem is assumed to be a quadratic program of form (2) as in Section 1.1. The columns of $-A^T$ are given the same names as the rows of A.

TRANSPOSE COLUMNS ADDED TO MATRIX

$\begin{bmatrix} -A^T \\ \\ 0 \end{bmatrix}$ added to make M square.

EMPTY RØW 'row no.' ENCOUNTERED-ABANDONING PROBLEM

An empty row (number 'row no.') has been encountered in trying to construct $-A^T$. Empty columns are not allowed. Fatal error.

MATRIX SCALING CALLED FOR

IFSCAL was set equal to a non-zero value and M will be scaled.

INITIAL SUM OF SQUARES OF EXPONENTS = 'number'

Initial status of matrix before scaling by exponent.

NEW SUM OF SQUARES OF EXPONENTS = 'number'.

Status of matrix after scaling. All scaling is done on expcnents only, and hence scaling factors are powers of 16. If the matrix was well scaled to begin with, the new sum of squares may be larger than the old.

SINGULAR M MATRIX-NO SCALING PERFORMED

The scaling routine detected a zero row or column. The run will continue but normal termination is unlikely.

DUMMY COLUMN WITH 'number' ELEMENTS CONSTRUCTED

There are negative elements in the q vector and a dummy column has been constructed for "DUMMY Z".

PROBLEM HAS TRIVIAL SOLUTION

Either q is non-negative or the given starting basis yields a feasible complementary solution. No dummy column is constructed and the solution is printed out.

STARTING BASIS WILL BE READ FROM FILE 'number'.

KINB was specified to be 'number'

BASIS GIVEN NOT FEASIBLE, WILL ENTER RECOVR

The given basis (complementary or almost complementary) is infeasible. The recovery routine is called to construct a new basis and dummy column.

WARNING - BASIS NAME CARD MISSING

WARNING-BASIS HAS NAME 'name'-DIFFERENT FROM PROBLEM

Warning messages on reading a basis. May not be the correct basis for the problem. Run continues.

NO MATCH FOR VECTOR

The basis file has a vector name not found in the problem. Vector ignored and run continues. Will almost certainly yield singular or infeasible basis and a call to the recovery routine.

NO MATCH FOR ROW 'rownam' TO SWAP WITH DUMMY Z

The row name 'rownam' cannot be found when reading the basis and the "DUMMY Z" variable is not put in the basis.

The run will continue.

SPURIOUS CARD IN BASIS FILE-'text'-ABORT

An unrecognizable card in the basis file - fatal error.

ERROR IN TRYING TO READ BASIS FROM FILE 'number'-CHECK JCL AND PARAMETERS-RUN ABORTED

A FORTRAN read error has occurred trying to read file KINB. Fatal error..

Run Time Messages

FEAS LOST ON ROW 'row no." VAR 'variable name' = 'value'

A loss of feasibility is detected when attempting to choose a pivot row. Call to RECOVR is triggered.

MATRIX SINGULAR-VECTOR 'column name' REMOVED

INVERT has detected a singularity. See below.

INVERT FAILS. WILL ATTEMPT TO RECOVR

One or more singularities were detected by INVERT. A call to RECOVR is triggered.

INSUFFICIENT ETA SPACE TO INVERT-RUN ABANDONED

INCREASE ETA SPACE AND NEMAX

The arrays IE(·) and E(·) in the program must be increased from their current dimension (standard value is NEMAX = 8000).

RELATIVE ERROR IN X = 'value'

Error monitoring message given every ITCH iterations (standard) is just before INVERT)

ITERATION LIMIT EXCEEDED

ITRLIM iterations have been performed. End of run.

BASIS WRITTEN ON FILE 'number' at ITERATION 'number'

The basis has been written on file KOUTB.

NONCOMPLEMENTARY PAIR NOT DEFINED, BUT DUMMY Z IN BASIS, UNRESOLVABLE ERROR-NO BASIS SAVED.

The solution is supposedly complementary but DUMMY Z is basic. Highly unlikely unless program corrupted.

Recovery Routine

FATAL ERROR-'number' BASIC VECTORS

The problem does not have NROW basic vectors. Probably due to corrupted program or split vectors.

RECOVERY FAILS-CANNOT FIND COMPLEMENTARY BASIS

Probably due to split vectors.

COMPLEMENTARY BASIS FOUND

RECOVR has found a new complementary basis.

FEAS COMP BASIS ARRIVED AT IN RECOVR

RECOVR has (fortuitously) found the complementary basis it has constructed to be feasible. Problem solved.

NEW DUMMY COLUMN CONSTRUCTED

RECOVR has constructed a new dummy column and determined a pivot row which will give a feasible almost complementary solution. Returns to normal algorithm.

4. SAMPLE RUNS

4.1. Problem Descriptions

Assume the user wishes to solve the following quadratic programming problems:

1) $\min x_1^2 \quad \text{s.t.} \quad 2x_1 + 3x_2 \le 6,\ 2x_1 + x_2 \le 4, \qquad x_1, x_2 \ge 0$,

2) $\min 2x_1^2 + 2x_1x_2 + 2x_2^2 + 2x_1x_3 + x_3^2 - 8x_1 - 6x_2 - 4x_3$

$\text{s.t.} \quad x_1 + x_2 + 2x_3 \le 3, \qquad x_1, x_2, x_3 \ge 0$,

3) $\min \frac{1}{2} x_1^2 - x_1x_2 + \frac{1}{2} x_2^2 - x_1 \qquad \text{s.t.} \quad x_1, x_2 \ge 0$.

First the user must formulate these as linear complementarity problems:

1) $$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{bmatrix} 2 & 0 & 2 & 2 \\ 0 & 0 & 3 & 1 \\ -2 & -3 & 0 & 0 \\ -2 & -1 & 0 & 0 \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 6 \\ 4 \end{pmatrix}$$

2) $$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{bmatrix} 4 & 2 & 2 & 1 \\ 2 & 1 & 0 & 1 \\ 2 & 0 & 2 & 2 \\ -1 & -1 & -2 & 0 \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} + \begin{pmatrix} -8 \\ -6 \\ -4 \\ 3 \end{pmatrix}$$

3) $$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

and put them in suitable MPS-type format as follows.

4.1. Program Input

```
 &PARAM NQUAD=2,&END
NAME              TEST1
ROWS
     W1
     W2
     W3
     W4
COLUMNS
     Z1          W1          2.0          W3          -2.0
     Z1          W4         -2.0
     Z2          W3         -3.0          W4          -1.0
     Z3          W1          2.0          W2           3.0
     Z4          W1          2.0          W2           1.0
RHS
     RHS         W3          6.0          W4           4.0
ENDATA
 &PARAM NQUAD=3,&END
NAME              TEST2
ROWS
     W1
     W2
     W3
     W4
COLUMNS
     Z1          W1          4.0          W2           2.0
     Z1          W3          2.0          W4          -1.0
     Z2          W1          2.0          W2           4.0
     Z2          W4         -1.0
     Z3          W1          2.0          W3           2.0
     Z3          W4         -2.0
     Z4          W1          1.0          W2           1.0
     Z4          W3          2.0
RHS
     RHS         W1         -8.0          W2          -6.0
     RHS         W3         -4.0          W4           3.0
ENDATA
 &PARAM NQUAD=2,&END
NAME              TEST3
ROWS
     W1
     W2
COLUMNS
     Z1          W1          1.0          W2          -1.0
     Z2          W1         -1.0          W2           1.0
RHS
     RHS         W1         -1.0          W2           0.0
ENDATA
```

4.2. Program Output

```
PROBLEM  TEST1

PROBLEM STATISTICS
     4 ROWS
     4 STRUCTURAL COLUMNS
     0 NON-ZERO ELEMENTS
DENSITY = 0.55250
PROBLEM HAS TRIVIAL SOLUTION
```

TEST1

'W' OR ROW VARIABLES		'Z' OR COLUMN VARIABLES	
VARIABLE	VALUE	VARIABLE	VALUE
W1	0.000000	Z1	0.000000
W2	0.000000	Z2	0.000000
W3	6.000000	Z3	0.000000
W4	4.000000	Z4	0.000000

QUADRATIC OBJECTIVE FUNCTION VALUE = 0.000000

PROBLEM TEST2

PROBLEM STATISTICS

4 ROWS

4 STRUCTURAL COLUMNS

13 NON-ZERO ELEMENTS

DENSITY = 0.81250

DUMMY COLUMN WITH 3 ELEMENTS CONSTRUCTED

INVERT STATISTICS
6 NONZ IN BASIS
1 STRUCTURAL COLUMNS IN BASIS
0 VECTORS ABOVE BUMP
4 VECTORS BELOW BUMP
L: 0 NONZ 0 ETAS
U: 3 NONZ 1 ETAS
TOTALS: 2 OFF DIAG NONZ 1 ETAS

RELATIVE ERROR IN X = 0.00000E 00

ITCOUNT	STATUS	DUMMY Z	VECIN	VECOUT	NETA	NELEM
0	A	8.00000000	DUMMY Z	W1	1	3
1	A	4.00000000	Z1	W2	1	3
2	A	1.00000000	Z2	W3	2	7
3	A	0.50000000	Z3	W4	3	11
4	A	0.00000000	Z4	DUMMY Z	4	15
4	C	0.00000000	Z4	DUMMY Z	4	15

TEST2

'W' OR ROW VARIABLES VARIABLE	VALUE	'Z' OR COLUMN VARIABLES VARIABLE	VALUE
W1	0.000000	Z1	1.333333
W2	0.000000	Z2	0.777778
W3	0.000000	Z3	0.444444
W4	0.000000	Z4	0.222222

QUADRATIC OBJECTIVE FUNCTION VALUE = -8.888889

PROBLEM TEST3

PROBLEM STATISTICS

2 ROWS

2 STRUCTURAL COLUMNS

4 NON-ZERO ELEMENTS

DENSITY = 1.00000

DUMMY COLUMN WITH 1 ELEMENTS CONSTRUCTED

INVERT STATISTICS
2 NONZ IN BASIS
1 STRUCTURAL COLUMNS IN BASIS
0 VECTORS ABOVE BUMP
2 VECTORS BELOW BUMP
L: 0 NONZ 0 ETAS
U: 1 NONZ 1 ETAS
TOTALS: 0 OFF DIAG NONZ 1 ETAS

RELATIVE ERROR IN X = 0.00000E 00

ITCOUNT	STATUS	DUMMY Z	VECIN	VECOUT	NETA	NELEM
0	A	1.00000000	DUMMY Z	W1	1	1
1	A	1.00000000	Z1	W2	1	1
1	U	1.00000000	Z2	W2	2	3

TEST3

'W' OR ROW VARIABLES		'Z' OR COLUMN VARIABLES	
VARIABLE	VALUE	VARIABLE	VALUE
W1	0.000000	Z1	0.000000
W2	0.000000	Z2	0.000000
		DUMMY Z	1.000000

QUADRATIC OBJECTIVE FUNCTION VALUE = 0.000000

4.3. Use of the NQUAD Input Option

To illustrate use of the NQUAD input option for quadratic programs we reconsider problem TEST2 from the previous section. This is a QP in three variables with only one constraint. In Section 4.1 the problem is set up to include the entire M matrix including the $-A^T$ column from the constraint transpose. Below we enter only the first three columns of M. The fourth column is constructed internally and given the name "W4". This naming can cause some slight confusion in the iteration log, but it seems reasonable to name these dual variables after the rows to which they correspond.

Input Deck:

```
 &PARAM NQUAD=3,&END
NAME              TEST2
ROWS
    W1
    W2
    W3
    W4
COLUMNS
    Z1        W1          4.0         W2          2.0
    Z1        W3          2.0         W4         -1.0
    Z2        W1          2.0         W2          4.0
    Z2        W4         -1.0
    Z3        W1          2.0         W3          2.0
    Z3        W4         -2.0
RHS
    RHS       W1         -8.0         W2         -6.0
    RHS       W3         -4.0         W4          3.0
ENDATA
```

```
PROBLEM  TEST2


PROBLEM STATISTICS
     4 ROWS
     3 STRUCTURAL COLUMNS
    10 NON-ZERO ELEMENTS
DENSITY = 0.83333
NQUAD COLUMNS ONLY SUPPLIED - WILL ATTEMPT TO FORM      1 EXTRA COLUMNS OF CONSTRAINT MATRIX TRANSPOSE
TRANSPOSE COLUMNS ADDED TO MATRIX
DUMMY COLUMN WITH     3 ELEMENTS CONSTRUCTED

INVERT STATISTICS
    6 NONZ IN BASIS
    1 STRUCTURAL COLUMNS IN BASIS
    0 VECTORS ABOVE BUMP
    4 VECTORS BELOW BUMP
L:      0 NONZ      0 ETAS
U:      3 NONZ      1 ETAS
TOTALS:     2 OFF DIAG NONZ     1 ETAS

RELATIVE ERROR IN X =     0.000000E 00


ITCOUNT  STATUS       DUMMY Z          VECIN       VECOUT          NETA   NELEM
      0     A      0.000000000     DUMMY Z     W1                     1       3
      1     A      0.000000000     Z1          W2                     1       3
      2     A      1.000000000     Z2          W3                     2       7
      3     A      0.000000000     Z3          W4                     3      11
      4     A      0.000000000     W4          DUMMY Z                4      15
      4     C      0.000000000     W4          DUMMY Z                4      15
```

TEST?

'W' OR ROW VARIABLES		'Z' OR COLUMN VARIABLES	
VARIABLE	VALUE	VARIABLE	VALUE
W1	3.000000	Z1	1.333333
W2	0.000000	Z2	0.777778
W3	0.000000	Z3	0.444444
W4	0.000000	W4	0.222222

QUADRATIC OBJECTIVE FUNCTION VALUE = -8.888889

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

(14) 1. REPORT NUMBER: SOL-76-16 — | 2. GOVT ACCESSION NO. | (9) 3. RECIPIENT'S CATALOG NUMBER

(6) 4. TITLE (and Subtitle): USERS GUIDE FOR LCPL. A Program for Solving Linear Complementarity Problems by Lemke's Method. | 5. TYPE OF REPORT & PERIOD COVERED: Technical Report, | 6. PERFORMING ORG. REPORT NUMBER

(10) 7. AUTHOR(s): J. A. Tomlin | (15) 8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0865, DAAG29-74-C-0034

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Operations Research, Stanford University, Stanford, CA 94305 | (16) 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR-047-143

(11) 11. CONTROLLING OFFICE NAME AND ADDRESS: Operations Research Program Code 434, Office of Naval Research, Arlington, Virginia 22217

Mathematics Division, U.S. Army Research Office, Box CM, Duke Station, Durham, North Carolina 27706

12. REPORT DATE: August 1976 ✓

13. NUMBER OF PAGES: 33

14. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

(12) 36p.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Mathematical Programming
Linear Complementarity

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document is a users guide for LCPL, an efficient robust program for solving Linear Complementarity Problems by Lemke's method.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408 765

AB